

# Exploring Strategy-Proofness, Uniqueness, and Pareto Optimality for the Stable Matching Problem with Couples

Andrew Perrault and Joanna Drummond and Fahiem Bacchus

Department of Computer Science

University of Toronto

{perrault,jdrummond,fbacchus}@cs.toronto.edu

May 14, 2015

## Abstract

The Stable Matching Problem with Couples (SMP-C) is a ubiquitous real-world extension of the stable matching problem (SMP) involving complementarities. Although SMP can be solved in polynomial time, SMP-C is NP-Complete. Hence, it is not clear which, if any, of the theoretical results surrounding the canonical SMP problem apply in this setting. In this paper, we use a recently-developed SAT encoding to solve SMP-C exactly. This allows us to enumerate all stable matchings for any given instance of SMP-C. With this tool, we empirically evaluate some of the properties that have been hypothesized to hold for SMP-C.

We take particular interest in investigating if, as the size of the market grows, the percentage of instances with unique stable matchings also grows. While we did not find this trend among the random problem instances we sampled, we did find that the percentage of instances with an resident optimal matching seems to more closely follow the trends predicted by previous conjectures. We also define and investigate resident Pareto optimal stable matchings, finding that, even though this is important desideratum for the deferred acceptance style algorithms previously designed to solve SMP-C, they do not always find one.

We also investigate strategy-proofness for SMP-C, showing that even if only one stable matching exists, residents still have incentive to misreport their preferences. However, if a problem has a resident optimal stable matching, we show that residents cannot manipulate via truncation.

## 1 Introduction

Stable matching problems are ubiquitous, with many real-world applications, ranging from dating markets, to labour markets, to the school choice problem. Arguably the most well-known of these applications is the residency matching problem [Niederle *et al.*, 2008; Roth, 1984; Abdulkadiroglu *et al.*, 2005], instantiated in the large National Resident Matching Program (NRMP), among others. Importantly,

these markets have adapted to the needs of their participants over the years, allowing them to more richly express their preferences, or guarantee certain properties of the match valuable to participants [NRMP, 2013]. To address the problem of couples wanting to coordinate their placements, the NRMP began allowing couples to jointly express their preferences over residency programs. We call the resulting matching problem the Stable Matching Problem with Couples (SMP-C).

As with the standard SMP, the goal of SMP-C is to find a matching such that no pair (one from each side of the market) has an incentive to defect from their assigned placement. Such a matching is called *stable*. In their seminal paper on stable matching, Gale and Shapley provide a polytime Deferred Acceptance (DA) algorithm guaranteed to find a stable, male-optimal (resident-optimal) matching [1962]. They also proved many useful properties of SMP; e.g., a unique stable matching always exists, and a stable resident-optimal matching always exists (a matching is resident-optimal if no resident is better off in any other stable matching).

However, when a pair of doctors are allowed to jointly specify their rankings, many of these desirable properties no longer hold. In particular, SMP-C is NP-complete [Ronn, 1990]. Also, a stable matching might not exist. Even when it does exist a resident-optimal matching might not exist. SMP-C is not the only NP-complete extension of SMP. Previous literature on these other NP-complete extensions has investigated techniques that are not guaranteed to return stable matchings, such as local search techniques (e.g., [Marx and Schlotter, 2011; Gelain *et al.*, 2010]). Another approach has entailed encoding these NP-complete stable matching problems (e.g., the stable matching problem with incomplete lists and ties) into other NP-complete formalisms such as constraint satisfaction problems (CSP) or satisfiability (SAT) (e.g., [Gent and Prosser, 2002; Gent *et al.*, 2001; Unsworth and Prosser, 2005]). The advantage of such encodings is that effective solvers exist for CSP and SAT. However, these works do not address SMP-C.

Previous work on SMP-C has mainly focused on providing sound but not complete algorithms that extend the DA algorithm to deal with couples [Roth and Peranson, 1999; Kojima *et al.*, 2013]. A new SAT encoding for SMP-C was recently developed [Drummond *et al.*, 2015], showing very promising results when run on a competitive SAT solver

[Biere, 2013]. This encoding, SAT-E, scales fairly well, and is sometimes able to find stable matchings for problems on which the DA-style algorithms cannot find any. Importantly, this new SAT encoding allows for flexibility unavailable in DA-style algorithms: adding new constraints and objectives can be as simple as adding another constraint to the encoding.

In this paper, we use this flexibility to allow us to enumerate all possible stable matches of an instance of SMP-C. We also introduce a constraint that allows us to, given a stable solution to SMP-C, determine if that matching is resident Pareto optimal, and if it is not, find a resident Pareto optimal matching. This new machinery allows us to empirically evaluate interesting properties of SMP-C. In particular, we are interested in the following properties: when do instances of SMP-C have a unique matching? If an instance has multiple stable matchings, when is there a resident optimal matching? How does the existence of a stable matching change as the percentage of couples in the market increases?

Previous theoretical work has addressed some of these questions; Ashlagi *et al.* and Kojima *et al.* proved theoretical results regarding the guaranteed existence of a stable matching, if the size of the market grows sufficiently faster than the percentage of couples in the matching [2014; 2013]. Immorlica and Mahdian proved that, for SMP, and a fixed length ROL (with everyone drawing from some given distribution), the chance of drawing a problem with a unique stable matching goes to 1 as the size of the market goes to infinity. However, to our knowledge, no analogous results exist for SMP-C. We thus create our experiments in an attempt to simulate the settings in these papers.

Also, previous work has conjectured that some of the properties present in SMP may carry over to SMP-C. Roth and Peranson hypothesized that the reason that they see few opportunities for strategic behaviour in their data is that there are few stable matchings in large SMP-C markets [1999]. Furthermore, Roth and Peranson cite that strategy-proofness and resident optimality are desirable properties for stable matchings.

In this paper, we first provide a theoretical contribution, showing that, for SMP-C, we are not guaranteed resident strategy-proofness, even in a problem with a unique stable matching. We do show, however, when a matching has resident optimal solution, residents cannot manipulate via truncation; they must manipulate via reordering their preferences. We thus show that, for SMP-C, uniqueness and resident optimality are both insufficient for strategy-proofness, but, in certain situations, it is “harder” for the residents to manipulate. We then provide two extensions to the already existing SAT-E encoding for SMP-C, allowing us to enumerate all stable matches and find a resident Pareto optimal match. We use these tools to empirically explore the space of SMP-C solutions. Our experiments suggest that Roth and Peranson’s observation that there were few instances where residents could improve by truncation is due to the high fraction of instances that have a resident optimal matching, a fraction that appears to increase with instance size.

## 2 Background

We cast our formalization of SMP-C (stable matching problem with couples) in terms of the well known residency matching problem, a labour market where doctors are placed into hospital residency programs [Roth and Peranson, 1999].

Informally, doctors wish to be placed into (matched with) some hospital program, and programs wish to accept some number of doctors. Both doctors and hospitals have preferences over who they are matched with; expressed as ranked order lists (ROLs). Some doctors are paired into couples, and these couples provide a joint ROL specifying their joint preferences. Both doctors and hospitals can provide incomplete lists, any alternative not listed is considered to be unacceptable. That is, they would rather not be matched at all than matched to an alternative not on their ROL. The SMP-C problem is to find a *stable* matching, such that no doctor-hospital pair has an incentive to defect from the assigned matching.

### 2.1 SMP-C

More formally, let  $D$  be a set of doctors and  $P$  be a set of programs. Since there is a preference to be unmatched over an unacceptable match, we use *nil* to denote this “doctor” or “program” alternative: matching a program  $p$  to *nil* indicates that  $p$  has an unfilled slot while matching a doctor  $d$  to *nil* indicates that  $d$  was not placed into any program. We use  $D^+$  and  $P^+$  to denote the sets  $D \cup \{\text{nil}\}$  and  $P \cup \{\text{nil}\}$  respectively.

The doctors are divided into two disjoint subsets,  $S \subseteq D$  and  $D \setminus S$ .  $S$  is the set of single doctors and  $D \setminus S$  is the set of doctors that are in couple relationships. We specify the couples by a set of pairs  $C \subseteq (D \setminus S) \times (D \setminus S)$ . If  $(d_1, d_2) \in C$  we say that  $d_1$  and  $d_2$  are each other’s partner. We require that every doctor who is not single (i.e., every doctor in  $D \setminus S$ ) have one and only one partner in  $C$ .

Each program  $p \in P$  has an integer quota  $\text{cap}_p > 0$ . This quota determines the maximum number of doctors  $p$  can accept (i.e.,  $p$ ’s capacity).

Everyone participating in the matching market has preferences over their alternatives. Each participant  $a$  specifies their preferences in a ROL which lists  $a$ ’s preferred matches from most preferred to least preferred. The ROLs of single doctors  $d \in S$  contain programs from  $P^+$ ; the ROLs of couples  $c \in C$  contain pairs of programs from  $P^+ \times P^+$ ; and the ROLs of programs  $p \in P$  contain doctors from  $D^+$ . Every ROL is terminated by *nil* (couple ROLs are terminated by  $(\text{nil}, \text{nil})$ ) since being unmatched is always the least preferred option, but is preferred to any option not on participant’s  $a$ ’s ROL.

The order of items on  $a$ ’s ROL defines a partial ordering relation where  $x \succeq_a y$  indicates that  $x$  appears before  $y$  on  $a$ ’s ROL or  $x = y$ . We define  $\succ_a$ ,  $\preceq_a$ , and  $\succsim_a$  in terms of  $\succeq_a$  and equality in the standard way. We say that  $x$  is *acceptable* to  $a$  if  $x \succeq_a \text{nil}$ . (Note that unacceptable matches are not ordered by  $\succeq_a$ .)

We define a choice function  $\text{Ch}_p()$  for programs  $p \in P$ . Given a set of doctors  $R$ ,  $\text{Ch}_p(R)$  returns the subset of  $R$  that  $p$  would prefer to accept.  $\text{Ch}_p(R)$  is the maximal subset of  $R$  such that for all  $d \in \text{Ch}_p(R)$ ,  $d \succ_p \text{nil}$ , for all

$d' \in R - Ch_p(R)$ ,  $d \succ_p d'$ , and  $|Ch_p(R)| \leq cap_p$ . It is convenient to give the null program a choice function as well:  $Ch_{nil}(O) = O$ , i.e.,  $nil$  will accept any and all matches.

We use the notation  $ranked(a)$  to denote the set of options that  $a$  could potentially be matched with. For single doctors  $d$  and programs  $p$  this is simply the ROLs of  $d$  ( $rol_d$ ) and  $p$  ( $rol_p$ ). For a doctor that is part of a couple  $(d_1, d_2)$ ,  $ranked(d_1) = \{p_1 | \exists p_2. (p_1, p_2) \in rol_{(d_1, d_2)}\} \cup \{nil\}$  and similarly for  $d_2$ . Note that  $nil \in ranked(a)$ .

Finally, we use the function  $rank_a(x)$  to find the index of match  $x$  in  $a$ 's  $rol$ :  $rank_a(x) = i$  iff  $x$  appears at index  $i$  (zero-based) on  $a$ 's ROL,  $rol_a$ , or  $|rol_a|$  if  $x$  does not appear on  $rol_a$ . Also we use  $rol_a$  as an indexable vector, e.g., if  $x$  is acceptable to  $a$  then  $rol_a[rank_a(x)] = x$ .

## 2.2 Stable Matchings

**Definition 1** (Matching). A **matching**  $\mu$  is a mapping from  $D$  to  $P^+$ . We say that a doctor  $d$  is matched to a program  $p$  under  $\mu$  if  $\mu(d) = p$ , and that  $p$  is matched to  $d$  if  $d \in \mu^{-1}(p)$ .

We want to find a matching where no doctor-program pair has an incentive to defect. We call the pairs that do have an incentive to defect blocking pairs. First we define the condition  $willAccept(p, R, \mu)$  to mean that program  $p$  would prefer to accept a set of doctors  $R$  over its current match  $\mu^{-1}(p)$ :  $willAccept(p, R, \mu) \equiv R \subseteq Ch_p(\mu^{-1}(p) \cup R)$ . Note that if  $p$  is already matched to  $R$  under  $\mu$  then  $p$  will accept  $R$ .

- Definition 2** (Blocking Pairs for a Matching  $\mu$ ). 1. A **single doctor**  $d \in S$  and a program  $p \in P$  is a **blocking pair** for  $\mu$  if and only if  $p \succ_d \mu(d)$  and  $willAccept(p, \{d\}, \mu)$ .
2. A couple  $c = (d_1, d_2) \in C$  and a program pair  $(p_1, p_2) \in P^+ \times P^+$  with  $p_1 \neq p_2$  is a **blocking pair** for  $\mu$  if and only if  $(p_1, p_2) \succ_{(d_1, d_2)} (\mu(d_1), \mu(d_2))$ ,  $willAccept(p_1, \{d_1\}, \mu)$ , and  $willAccept(p_2, \{d_2\}, \mu)$ .
3. A couple  $c = (d_1, d_2)$  and a program  $p \in P$  is a **blocking pair** for  $\mu$  if and only if  $(p, p) \succ_{(d_1, d_2)} (\mu(d_1), \mu(d_2))$  and  $willAccept(p, \{d_1, d_2\}, \mu)$ .

**Definition 3** (Individually Rational Matching). A matching  $\mu$  is **individually rational** if and only if (a) for all  $d \in S$ ,  $\mu(d) \succeq_d nil$ , (b) for all  $c = (d_1, d_2) \in C$ ,  $(\mu(d_1), \mu(d_2)) \succeq_c (nil, nil)$ , and (c) for all  $p \in P$ ,  $|\mu^{-1}(p)| \leq cap_p$  and for all  $d \in \mu^{-1}(p)$  we have that  $d \succeq_p nil$ .

**Definition 4** (Stable Matching). A matching  $\mu$  is **stable** if and only if it is individually rational and no blocking pairs for it exist.

## 2.3 Residence Preferred Matchings

The set of stable matchings can be quite large. When there are no couples (i.e., in SMP) this set is always non-empty [Gale and Shapley, 1962] and has a nice structure: it forms a lattice under the partial order  $\succeq_R$  defined below [Knuth, 1976].

**Definition 5** (Residence Preferred Matchings). A matching  $\mu_1$  is **resident preferred** to another  $\mu_2$  when for all  $a \in S \cup C$  we have that  $\mu_1(a) \succeq_a \mu_2(a)$ . We denote this relationship as  $\mu_1 \succeq_R \mu_2$ . We also define  $\succ_R$ ,  $\prec_R$ , and  $\preceq_R$  in terms of  $\succeq_R$  and equality in the standard way. When  $\mu_1 \succ_R \mu_2$  we say that  $\mu_1$  **dominates**  $\mu_2$ .

In other words,  $\mu_1 \succ_R \mu_2$  if every doctor and couple gets the same or a better match in  $\mu_1$  as in  $\mu_2$  and at least one doctor or couple gets a better match.

The lattice structure and the fact there can only be a finite number of matchings means that for SMP there is always a resident-optimal matching.

**Definition 6** (Resident Optimal Matching). We say that a matching  $\mu$  is **resident optimal**, written  $\mathcal{R}_{opt}(\mu)$  when  $\mu$  is stable and  $\mu \succeq_R \mu'$  for all other stable matchings  $\mu'$  ( $\mu$  dominates all other stable matchings).

Note that, we restrict the resident-optimal matching to be stable. It can be observed that for any resident  $r$  (couple  $(r_1, r_2)$ )  $\mu(r)$  ( $(\mu(r_1), \mu(r_2))$ ) is the best match offered by any stable matching when  $\mathcal{R}_{opt}(\mu)$ .

Resident-optimality is generally cited as an important property for stable matching algorithms (e.g., [Gale and Shapley, 1962; Roth and Peranson, 1999]). In the presence of couples however, stable matchings might not exist and even when they do there have no lattice structure under  $\succeq_R$ . However, the  $\succeq_R$  is still well defined and for SMP-C leads to potentially multiple *Pareto optimal* matchings.

**Definition 7** (Resident Pareto Optimal Matchings). We say that a matching  $\mu$  is **resident Pareto optimal**, written  $\mathcal{RP}_{opt}(\mu)$  when  $\mu$  is stable and there does not exist another stable matching  $\mu'$  such that  $\mu' \succ_R \mu$ .

It is easy to see that in SMP-C every stable matching  $\mu$  either is an  $\mathcal{RP}_{opt}$  or is dominated by an  $\mathcal{RP}_{opt}$  matching. This means that an SMP-C instance has an  $\mathcal{R}_{opt}$  matching if and only if it has a unique  $\mathcal{RP}_{opt}$  matching. As we will see in Section 5 it is often the case that SMP-C instances have more than one  $\mathcal{RP}_{opt}$  matching (and thus no  $\mathcal{R}_{opt}$  matching).

## 3 Strategy-Proofness and $\mathcal{R}_{opt}$ Existence for SMP-C

For an SMP instance the mechanism that returns the  $\mathcal{R}_{opt}$  matching in response to the stated preferences is strategy-proof with respect to the residents. That is, it is a dominant strategy for each resident to state their true preferences [Roth and Sotomayor, 1992].

To our knowledge, no strategy-proofness results exist for SMP-C. However, this result for SMP leads us to hypothesize that if we restrict our attention to SMP instances in which an  $\mathcal{R}_{opt}$  matching exists, then a mechanism that returns the  $\mathcal{R}_{opt}$  matching is strategy-proof. Unfortunately, this is not true; furthermore, the even stronger condition of there being a unique stable matching for a problem instance doesn't guarantee strategy-proofness for the residents (let alone the hospitals). However, we do show that when there is an  $\mathcal{R}_{opt}$  matching, residents have no incentive to misreport their preferences via truncation.

**Theorem 1.** Restricting to instances of SMP-C in which an  $\mathcal{R}_{opt}$  matching exists, let  $y()$  be a mechanism that maps from such an instance to the  $\mathcal{R}_{opt}$  matching. Then  $y$  is strategy-proof for residents if residents are only allowed to manipulate via truncations. However,  $y$  might not be strategy-proof if residents are allowed to manipulate via reordering.

*Proof.* Let  $\mu^*$  be the  $\mathcal{R}_{opt}$  matching.

Part I: Truncations are strategy-proof. Suppose resident or couple  $d$  truncates its preferences at program or pair of programs  $p$ . Let  $\Omega$  be the set of stable matchings before the truncation and let  $\Omega'$  be the set of stable matchings after the truncation. Because  $d$ 's preferences above the truncation point are the same after truncation, any matching  $\mu \in \Omega'$  where  $\mu(d) \neq nil$  (resp.  $\mu(d) \neq (nil, nil)$  for a couple  $d$ ) is in  $\Omega$ : the stability conditions contributed by other residents and couples are the same after  $d$ 's truncation. However,  $\Omega'$  may contain stable matchings where  $d$  is matched to  $nil$  that are not in  $\Omega$ . Thus  $\{\mu \in \Omega' : \mu(d) \neq nil\} \subseteq \Omega$ .

There are two cases: either 1)  $d$  truncates below  $\mu^*(d)$  or 2)  $d$  truncates at or above  $\mu^*(d)$ . In the former, it can be seen that  $\mu^*$  is a stable matching of  $\Omega'$  and the only matches in  $\Omega'$  not in  $\Omega$  have  $d$  matched to  $nil$ . Hence,  $\mu^*$  is an  $\mathcal{R}_{opt}$  matching for  $\Omega'$  and  $d$  cannot improve. In the latter,  $\Omega'$  consists of matchings where  $d$  is matched to  $nil$  because any  $\mu \in \Omega'$  such that  $\mu(d) \succ_d \mu^*(d)$  would be in  $\Omega$  and would contradict  $\mu^*$ 's  $\mathcal{R}_{opt}$  status. Hence,  $d$  cannot improve in this case either.

Part II: Reordering is not strategy-proof. We provide a counterexample. See Figure 1. The preferences provided above the line show doctors' and programs' true preferences. Couples are identified by their doctor-doctor pair, and give joint preferences as expected. Given these true preferences, only one stable matching exists (shown in the third column above the line). However, even though only one stable matching exists (and thus this matching is  $\mathcal{R}_{opt}$ ), the single doctor  $r_0$  has incentive to misreport preferences via reordering. Instead of reporting  $a \succ b \succ c \succ d$  ( $r_0$ 's true preferences)  $r_0$  can be matched to  $b$  instead of  $c$  by reporting  $b \succ d \succ c \succ a$ . (All other participants in the market report their true preferences.) This results in the matching seen in the third column below the line, a matching that is not stable under the original preferences, where single doctor  $r_0$  is better off than when they reported their true preferences.  $\square$

Roth and Peranson investigate strategy-proofness for both residents and hospitals empirically in the NRMP setting [1999]. They do this via truncations. In their setting, they do find that residents can benefit (albeit minimally) from truncating their true preferences. We then hypothesize that this is due either: i) the instances in their setting frequently do not contain an  $\mathcal{R}_{opt}$  matching, or ii) their algorithm (RP99) does not find the  $\mathcal{R}_{opt}$  matching even when it does exist. In Section 5 we will see that both of these situations can occur. That is, on many problems no  $\mathcal{R}_{opt}$  matching exists (the best one can do is an  $\mathcal{RP}_{opt}$  matching). And even when an  $\mathcal{R}_{opt}$  matching does exist, their algorithm might not find it, reporting some other stable matching instead.

## 4 Solving SMP-C

In this section we examine solution methods for finding stable matchings for SMP-C. The standard approach has been to find an extension of the deferred acceptance algorithm that can handle couples. However, these extensions are incomplete: they are unable to determine whether or not a stable matching exists, and even when a stable matching does exist they might not be able to find one.

Since SMP-C is known to be NP-Complete [Ronn, 1990] it is also possible to encode it as another NP-Complete problem. In particular, it can be encoded as a SAT (satisfiability) problem. The advantage of doing this is that there has been a great deal of work on SAT solvers and state-of-the-art SAT solvers are routinely able to solve SAT problems involving millions of clauses.<sup>1</sup> The advantage of encoding SMP-C to SAT and then using a SAT solver is that this approach is complete: given sufficient compute resources it will either find a stable matching or prove that none exists.

### 4.1 Existing DA Algorithms for SMP-C

The basic principle of DA algorithms [Gale and Shapley, 1962] is that members of one side of the market propose down their ROLs while the other side either rejects those proposals or holds them until they see a better proposal: once all proposals have been made that side accepts the proposal they have not rejected (acceptance of proposals is deferred until the end).

Roth and Peranson develop a DA algorithm, RP99, capable of dealing with couples [1999]. This well known algorithm has been used with considerable success in practice, including most famously for finding matches for the NRMP which typically involves about 30,000 doctors [NRMP, 2013]. Using the description in [Roth and Peranson, 1999] we have implemented RP99. RP99 employs an iterative scheme. After computing a stable matching for all single doctors, couples are added one at a time and a new stable matching computed after each addition. The algorithm uses DA at each stage to find these stable matchings. Matching a couple can make previously made matches unstable and in redoing these matches the algorithm might start to cycle. Hence, cycle checking (or a timeout) is sometimes needed to terminate the algorithm.

Kojima et al. develop a simple "sequential couples algorithm" [2013]. This algorithm is analyzed to prove that the probability of a stable matchings existing goes to one under certain assumptions. However, this simple algorithm is not useful in practice as it declares failure under very simple conditions. Kojima et al. also provide a more practical DA algorithm, KPR, that they use in their experiments. We have implemented KPR. The main difference between KPR and RP99 is that KPR deals with all couples at the same time—it does not attempt to compute intermediate stable matchings. Interestingly, this makes KPR much more successful (and efficient) in our experiments.

Finally, Ashlagi et al. extend the analysis of Kojima et al., developing a more sophisticated "Sorted Deferred Acceptance Algorithm" and analyzing its behaviour [2014]. This algorithm is designed mainly to be amenable to theoretical analysis rather than for practical application. We have not implemented this algorithm so we do not include any empirical results about its performance.

### 4.2 Solving SMP via SAT

We have developed an effective encoding of SMP-C into SAT. This encoding and its performance is reported on in [Drum-

<sup>1</sup>Of course, since SAT is NP-Complete these solvers can also be foiled by small SAT instances. Nevertheless, they often exhibit very good performance on instances that arise from real world problems.

Doctor preferences	Program preferences	Unique stable matching
$r_0 : a \succ b \succ c \succ d$	$a : r_3 \succ r_0 \succ r_1$	$\mu(r_0) = c$
$(r_1, r_2) : (b, e) \succ (a, d)$	$b : r_1 \succ r_0$	$\mu((r_1, r_2)) = (b, e)$
$(r_3, r_4) : (a, d) \succ (c, e)$	$c : r_3 \succ r_0$	$\mu((r_3, r_4)) = (a, d)$
	$d : r_0 \succ r_2 \succ r_4$	
	$e : r_4 \succ r_2$	

---

Doctor preferences	Program preferences	Unique stable matching
$r_0 : b \succ d \succ c \succ a$	$a : r_3 \succ r_0 \succ r_1$	$\mu(r_0) = b$
$(r_1, r_2) : (b, e) \succ (a, d)$	$b : r_1 \succ r_0$	$\mu((r_1, r_2)) = (a, d)$
$(r_3, r_4) : (a, d) \succ (c, e)$	$c : r_3 \succ r_0$	$\mu((r_3, r_4)) = (c, e)$
	$d : r_0 \succ r_2 \succ r_4$	
	$e : r_4 \succ r_2$	

Figure 1: Counterexample for Strategy-Proofness in the Reordering Case

mond *et al.*, 2015].<sup>2</sup>

We call this encoding SAT-E, and given an SMP-C instance  $\langle D, C, P, ROLs \rangle$ , where  $ROLs$  is the set of all participant ROLs, SAT-E( $\langle D, C, P, ROLs \rangle$ ) can be viewed as a function that returns a SAT encoding in CNF (conjunctive normal form), which is the input format taken by modern SAT solvers.

There are three things to know about SAT-E.

1. For any SMP-C instance  $\mathcal{I} = \langle D, C, P, ROLs \rangle$ , the satisfying models of SAT-E stand in a one-to-one correspondence with the stable models of  $\mathcal{I}$ . This allows us to enumerate all stable models (see below).
2. SAT-E includes the set of propositional variables  $m_d[p]$  one for each  $d \in D$  and  $p \in D$  such that  $p \succeq_d nil$ . In any satisfying model,  $\pi$ ,  $m_d[p]$  is true if and only if  $\mu(d) = p$  in the stable matching  $\mu$  corresponding to  $\pi$ .
3. SAT-E also includes the set of propositional variables  $m_c[i]$  for each  $c \in C$  and  $i$  in the range  $[0, |rol_c - 1|]$  where  $rol_c$  is  $c$ 's ROL. In any satisfying model,  $\pi$ ,  $m_c[i]$  is true if and only if in  $\mu$ , the stable matching corresponding to  $\pi$ ,  $c$  is matched to a program pair they rank  $i$  or above in their ROL.

Given SAT-E and the above three facts we can develop two simple yet powerful algorithms: one for enumerating all stable models and the other for finding a Pareto Optimal matching, an  $\mathcal{RP}_{opt}$  matching, that dominates a stable matching  $\mu$ .

Algorithm 1 is our algorithm for enumerating all stable models. It uses a sequence of calls to a SAT solver to do this. It first constructs the SAT-E encoding of the SMP-C instance then enters a loop. Each time through the loop a new stable model is found by the SAT solver, and a *blocking clause*  $c$

<sup>2</sup>For the reader's convenience we have included a description of this encoding in the appendix. See [Drummond *et al.*, 2015] for full details.

**ALGORITHM 1:** Enumerate all stable models of an inputted SMP-C instance

**Input:**  $\mathcal{I} = \langle D, C, P, ROLs \rangle$  an SMP-C instance

**Output:** Enumerate all stable models of a  $\mathcal{I}$

```

1 CNF  $\leftarrow$  SAT-E( $\mathcal{I}$ )
2 while true do
3    $(sat?, \pi) \leftarrow$  SatSolve(CNF)
   /* SatSolve returns the status (sat or unsat) and a satisfying
   model  $\pi$  if sat */
4   if sat? then
5      $\mu \leftarrow$  stable matching corresponding to  $\pi$ 
6      $c \leftarrow \{\neg m_d[p] \mid \mu(d) = p\}$ 
7     CNF  $\leftarrow$  CNF  $\cup \{c\}$ 
8     enumerate( $\mu$ )
9   else
10    return // All stable matchings enumerated.
```

is added to the SAT-E encoding. This clause  $c$  is a disjunction that says that no future solution is allowed to return the same stable model (one of the mappings  $\mu(d) = p$  of the corresponding stable model must be different, i.e., one of the variables  $m_d[p]$  made true by  $\mu$  must be false in every future matching).

Algorithm 2 is our algorithm for finding a  $\mathcal{RP}_{opt}$  matching that dominates an inputted stable matching  $\mu$ . The algorithm takes an SMP-C instances as input and constructs the SAT-E encoding for that instance. It then blocks the match  $\mu$  from being a satisfying solution (using the same kind of clause as Algo. 1) and also forces the next match found to be  $\succeq_{\mathcal{R}}$  the current match. It accomplishes this by adding a clause for each single doctor  $d$  that says that  $d$  must be matched to a program it ranks at least as high as  $\mu(d)$ , and for every couple  $c$  a (unit) clause that says that  $c$  must be matched to a pair of programs it ranks at least as highly as  $\mu(c)$ . This causes the new match to be  $\succeq_{\mathcal{R}}$  to the current match, and since the new match cannot be equal it must be  $\succ_{\mathcal{R}}$  the current match. That

**ALGORITHM 2:** Given an SMP-C instance and a stable matching  $\mu$  find a dominating  $\mathcal{RP}_{opt}$  matching.

**Input:**  $\mathcal{I} = \langle D, C, P, ROLs \rangle$  an SMP-C instance and  $\mu$  a stable matching for  $\mathcal{I}$

**Output:** An  $\mathcal{RP}_{opt}$  matching for  $\mathcal{I}$  that dominates  $\mu$

```

1 while true do
2   CNF  $\leftarrow$  SAT-E( $\mathcal{I}$ )
3    $c \leftarrow \{\neg m_d[p] \mid \mu(d) = p\}$ 
4   CNF  $\leftarrow$  CNF  $\cup \{c\}$ 
5   for  $d \in S$  do
6      $c_d \leftarrow \{m_d[p] \mid p \succeq_d \mu(d)\}$ 
7     CNF  $\leftarrow$  CNF  $\cup \{c_d\}$ 
8   for  $c \in C$  do
9      $c_c \leftarrow \{m_c[i] \mid i = rank_c(\mu(c))\}$ 
10    CNF  $\leftarrow$  CNF  $\cup \{c_c\}$ 
11  (sat?,  $\pi$ )  $\leftarrow$  SatSolve(CNF)
12  if sat? then
13     $\mu \leftarrow$  stable matching corresponding to  $\pi$ 
14  else
15    return  $\mu$  // Return last match found.
```

is, the new match must dominate the current match. If no dominating match can be found, the current match is  $\mathcal{RP}_{opt}$  and we return it.

## 5 Empirical Results

In this section we report on various experiments we performed. We used the state-of-the art SAT solver Lingeling [Biere, 2013] to solve our encoding SAT-E; Algorithm 1 to find all stable matches; and our implementation of the two DA algorithms RP99 and KPR, described in Section 4.1, when testing the effectiveness of DA algorithms.

### 5.1 Statistical models

We experiment with randomly generated SMP-C instances. In our experiments we confine our attention to instances in which all program quotas are 1. So these are one-to-one matching problems with couples.

We generate ordered lists from sets using a common random sampling procedure. To obtain an ordered list  $L$  of size  $k$  from a set  $S$ , given that we already generated the first  $i$  items, we draw elements from  $S$  independently and uniformly at random until we find an element  $e$  that does not already appear in  $L$ . Once we have selected such an  $e$  it becomes the  $i + 1$ 'th item of  $L$ . This process stops when  $L$  has  $k$  items.

We use market sizes of size  $n$  where  $n$  ranges from 200 to 20,000, and varying percentages  $x$  of couples. For each  $n$  we include  $n$  doctors in  $D$  and  $n$  programs in  $P$ . Among the doctors we mark  $1 - x$  percent as being singles and the remaining  $x$  percent are paired into couples. (Thus we have  $x * n/2$  couples and  $n - x * n$  singles). For each single we randomly generate an ordered ROL of length 5 from  $P$  (using the procedure described above), each couple has an ROL of length 15 randomly generated from  $P^+ \times P^+ - \{(nil, nil)\}$ , and each program has an ROL that includes all doctors that ranked it (including one member of a couple) and is randomly generated from that same set. For each setting of the parameters, we drew 50 problem instances.

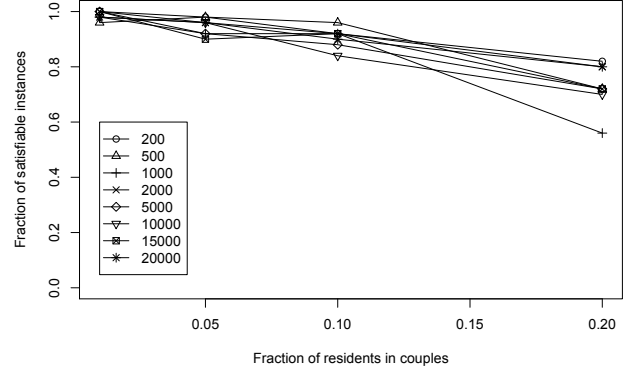


Figure 2: Fraction of satisfiable instances drawn, varying the fraction of couples in the match, and the size of the market.

### 5.2 Experiments

#### Existence of a Stable Matching in SMP-C

Our first experiment involves running a SAT solver on each SMP-C instance. This allows us to determine how often these instances had a stable match. Figure 2 shows the trend. We see that as the fraction of residents who are in couples increases, the fraction of satisfiable instances drops fairly rapidly. Interestingly, it appears as though this effect is independent of the size of the market; for all market sizes drawn, the fraction of satisfiable instances is fairly consistent, and follows the same trend. These results agree with the predictions of Ashlagi et al. who showed that in a random model similar but not identical to ours whenever the number of couples grows linearly with market size, the probability of no stable matching existing is a constant. This agrees with our data that market size has little effect. Their model was not, however, able to predict what that probability might be, nor how it might change as the percentage of couples increases.

It can also be noted that the results presented here (20,000 couples with an average single's ROL of size 5) are of a size comparable to the NRMP, where there are 34,355 residents with an average single's ROL of size 11 [NRMP, 2013]. On NRMP Roth and Peranson [1999] remark that historically no instance was found not to have a stable matching. This might arise from (a) the percentage of couples being very small for NRMP, or (b) extra structure in this real problem not present in our random problems. If the cause is (a) there might be practical concerns for the ability of clearinghouse mechanisms to find stable matchings if the percentage of couples rises.

#### Existence of a Unique Stable Matching in SMP-C

Figure 3 shows the fraction of instances that had a unique stable matchings as the size of the market increases, and as the percentage of the couples in the market increases. Importantly, this figure only counts the fraction of instances with a unique matching among those instances that have at least one matching. This data was generated by running Algo. 1.

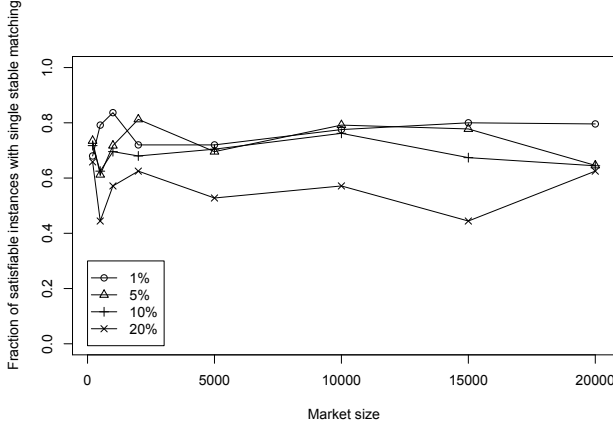


Figure 3: Fraction of satisfiable instances with single stable matching, varying size of the market and percentage of couples in the market.

As shown in Figure 2, the fraction of instances where a stable matching exists changes as the percentage of couples in the market changes. So to remove that underlying trend, we only look instances where a stable matching is known to exist.

Note that, while these curves generally mimic the pattern we’d expect to see (the fraction of unique matchings tends to increase as the market size gets larger), the curve appears to level out and reach an asymptote much earlier than expected. In our experiments, no setting reached more than 84% unique matchings.

Immorlica and Mahdian proved that, for SMP (where at least one stable matching is always true) with fixed length ROLs (and everyone drawing from some given distribution), the chance of drawing a problem with only one stable matching goes to 1 as the size of the market goes to infinity [2005].

Our trends for SMP-C, however, appear somewhat asymptotic, especially when the percentage of couples is low. It thus appears that, at least for the size of problems relative to the size of ROL investigated here, that the Immorlica and Mahdian results do not apply. This could be because their results give very large bounds that require the size of the problems to be much larger than drawn here. So our results might not be on sufficiently large markets to capture the phenomenon they describe. Alternately, this could be because their results do not hold for SMP-C.

As an aside, when comparing the fraction of unique satisfiable instances out of all problem instances drawn, the numbers look even worse; with a market size of 20,000 and 20% couples, the percentage of problems drawn with only one stable matching drops to 50%.

### Properties of the Set of Matchings for SMP-C

We next investigate the properties of the set of matchings of each SMP-C instance as generated by Algorithm 1. We have shown that, at least under the settings explored in this paper, unique stable matches are not as common as we may have hypothesized. We also have proven that, even if a unique

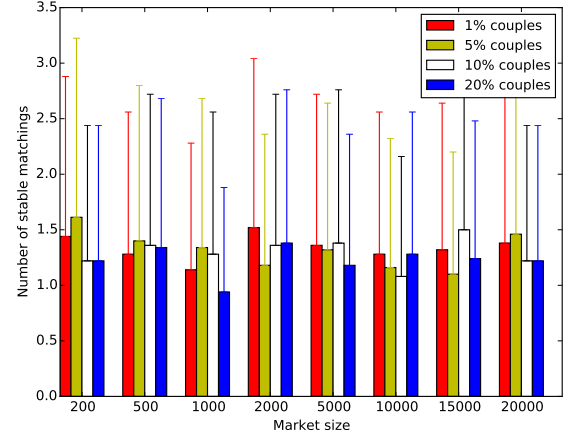


Figure 4: Number of stable matchings versus market size and percentage of couples present in the market. Box represents average number of stable matchings; whisker represents one standard deviation

matching exists (and hence it is an  $\mathcal{R}_{opt}$  matching), that is no guarantee of strategy-proofness on either the residents’ or programs’ side for SMP-C.

Nevertheless, we have also shown that if residents are only allowed to truncate their preferences (which is arguably the easiest misreporting strategy), then they have no incentive to misreport if an  $\mathcal{R}_{opt}$  matching exists. Since an  $\mathcal{R}_{opt}$  matching can exist even when there more than one stable matching, this can be a useful situation that covers more cases than having only one stable matching.

Figure 4 shows our results about this question. The boxes in this plot represents the average number of stable matches for different problem parameters; the whisker represents one standard deviation. For the problems we investigated, the average number of stable matchings was quite low (1.30), and did not seem to be affected much by either the size of the market, or the percentage of couples involved in the market. Note that exceedingly few problem instances have more than 3 stable matches, no matter how the parameters are set. Also, in each setting of the parameters, problem instances with one and two stable matchings are very common. Note that, unlike the previous analysis presented in Figure 2, we are looking at *all* problem instances, not just the satisfiable ones.

In the problems we investigated, 10.63% had no stable matching, 61.91% had a unique stable matching, 21.89% had two stable matchings, 0.31% had three stable matchings, 4.69% had four stable matchings, 0.13% had six stable matchings, and 0.44% had eight stable matchings.

When a problem instance has more than one stable matching, we are particularly interested in how many  $\mathcal{RP}_{opt}$  matchings exist. If only one  $\mathcal{RP}_{opt}$  matching exists this matching is also an  $\mathcal{R}_{opt}$  matching, and our theoretical result applies: residents have no incentive to misreport their preferences by truncation. 78.5% of all problem instances drawn had an  $\mathcal{R}_{opt}$  matching, which is many more instances than simply had a single stable matching (61.91%). Addition-

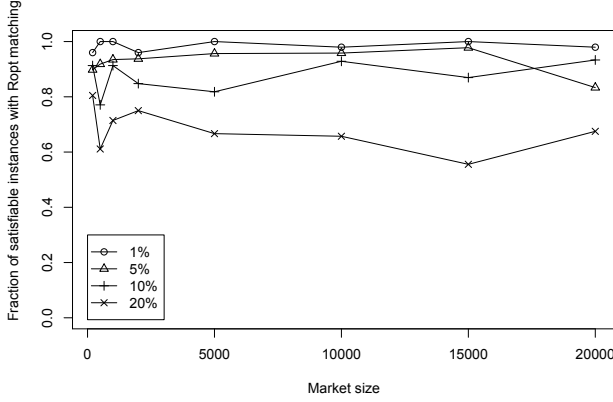


Figure 5: Fraction of satisfiable instances with an  $\mathcal{R}_{opt}$  matching, varying size of the market and percentage of couples in the market.

ally, if we again look at the Immorlica and Mahdian results, but with respect to fraction of satisfiable instance that had an  $\mathcal{R}_{opt}$  matching instead of the more restrictive condition of having only one stable matchings, we see a trend much closer to what we might expect. See Figure 5. Furthermore, many instances with a low percentage of couples almost always had an  $\mathcal{R}_{opt}$  matching; when the market had 1% couples, 98.5% of all satisfiable instances had an  $\mathcal{R}_{opt}$  matching; with 5%, 92.7% had an  $\mathcal{R}_{opt}$  matching. Importantly, this means that for many of the problem instances that we drew, residents have no incentive to misreport their preferences via truncation. However, further investigation is required to ascertain the relationship between the percentage of couples in a matching and the fraction of instances that have an  $\mathcal{R}_{opt}$  matching.

Finally, looked at those instances with more than one stable matching to see how many, on average,  $\mathcal{R}_{opt}$  matchings they contain. We found that in some cases  $\mathcal{R}_{opt}$  matchings are quite common. When there are two stable matchings for a given problem instance, there were an average of 1.36  $\mathcal{R}_{opt}$  matchings per problem. Instances with 3 stable matchings always had 3  $\mathcal{R}_{opt}$  matchings per problem. Instances with 4 stable matchings had an average of 1.69  $\mathcal{R}_{opt}$  matchings per problem, and instances with 8 stable matchings had an average of 4  $\mathcal{R}_{opt}$  per problem.

### 5.3 Structure of SMP-C Problem Instances

In this section we examine some finer grained structure among the set of stable matchings. We examined the stable matchings and a basis for the set of Pareto improving moves. A Pareto improving move is the set of resident transfers needed to transform matching  $\mu_2$  to matching  $\mu_1$  when  $\mu_1 \succ_{\mathcal{R}} \mu_2$  (i.e., the transfers improve the match). Improving moves that are the result of combining other improving are not counted as part of the basis.

We show this structure as a directed graph, where nodes are stable matchings for a given instance, and an edge goes from node  $a$  to  $b$  if there is a Pareto improving move transforming

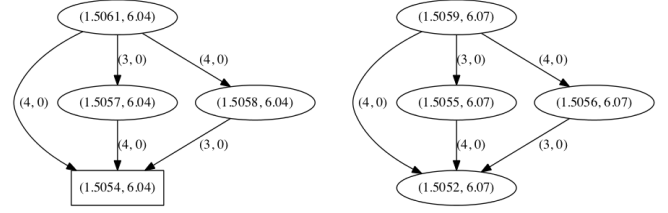


Figure 6: Pareto Improvement graph for one problem instance, with 8 stable matchings, and 2  $\mathcal{R}_{opt}$  matchings. Square denotes the matching KPR and RP99 found.

$a$  to  $b$ . (Note this implies that  $b \succ_{\mathcal{R}} a$ . Note this graph is not guaranteed to be connected, and in practice, many of these graphs have multiple components. By definition, each component will contain at least one  $\mathcal{R}_{opt}$  matching. While it appears to be exceedingly rare for any component to contain more than one component (at least in the problems we generated), we did find one component with two  $\mathcal{R}_{opt}$  matchings.

For an illustration of what these components and the possible Pareto improving moves may look like, see Figure 6. For the problem instance chosen, there were 8 stable matchings, 2 of which were  $\mathcal{R}_{opt}$  matchings. Each node in the graph contains a tuple of the singles' and couples' average rank; each edge on the graph is the maximum improvement for any single and any couple from the previous match to the new Pareto improved match.

Also note that in this problem instance, on average, one of these  $\mathcal{R}_{opt}$  matchings is better for couples (by 0.03 positions on average) and one of these  $\mathcal{R}_{opt}$  matchings is better for singles (by 0.0002 positions on average). Interestingly, there are very few unique Pareto optimal moves in this graph; there are only two. This tends to hold in general; when there are two stable matchings, there are an average of 0.64 unique Pareto moves per problem. With 4 stable matchings, there are an average of 1.48 Pareto moves.

Note that, for the single that has the greatest incentive to go from one matching to another, they have quite a high incentive—almost their entire ROL. (Remember, singles give a ROL of size 5.) Note that Figure 6 shows a case where only single residents improve; this is not true in the general case. Couples, likewise, can greatly benefit from moving from a non-Pareto optimal matching to a  $\mathcal{R}_{opt}$  matching. Additionally, some singles and couples will strongly prefer one  $\mathcal{R}_{opt}$  matching over another; in this example, the maximum incentive for any single to switch from the  $\mathcal{R}_{opt}$  matching on the left to the one on the right is moving up their ROL 3 positions. In this example, there is no incentive for couples to switch from the  $\mathcal{R}_{opt}$  matching on the left to the one on the right. For switching from the  $\mathcal{R}_{opt}$  matching on the right to the one on the left, the maximum incentive for any single to move is one position in their ROL, and 3 positions for any couple. In this instance, both of the DA algorithms KPR and RP99 found the same  $\mathcal{R}_{opt}$  matching, denoted by the square node in the graph. While KPR and RP99 are not guaranteed to find an  $\mathcal{R}_{opt}$  matching, they did in this instance.

While in this example, there is little incentive to switch from one  $\mathcal{R}_{opt}$  matching to another, there were instances



% Couples	% $\mathcal{RP}_{opt}$ found		% Total Timeouts	
	KPR	RP99	KPR	RP99
1	100.00	100.00	0.00	0.00
5	94.54	96.16	0.25	0.75
10	88.66	89.58	0.25	6.25
20	91.42	90.24	5.57	36.26

Table 1: DA algorithms performance versus percentage of couples in a match. For %  $\mathcal{RP}_{opt}$  found, only problems with more than one stable matching that did not timeout were considered. For % Total timeouts, all problems with at least one stable matching were considered.

in the data where at least one single could move up their *entire* preference list by switching from one  $\mathcal{RP}_{opt}$  matching to another, and at least one couple could move up their *entire* preference list by switching from one  $\mathcal{RP}_{opt}$  matching to another.

#### 5.4 The Performance of DA Algorithms for SMP-C

We are also interested in how the standard DA-style algorithms perform with respect to finding  $\mathcal{RP}_{opt}$  matchings. As Roth and Peranson mention that resident optimality is an important design aspect for stable matching algorithms, we investigate how frequently the DA algorithms find a  $\mathcal{RP}_{opt}$  matching. We do this by using Algo 1 to enumerate all possible stable matchings for each problem instance. We then identify all  $\mathcal{RP}_{opt}$  matchings, and then see if the DA algorithms find a  $\mathcal{RP}_{opt}$  matching for that instance. Table 5.4 shows these results. We did not find much of a relationship between the size of the problem and how frequently the DA algorithms found an  $\mathcal{RP}_{opt}$  matching, so we combine all problem sizes for this analysis. Note that, when analyzing the % of times the DA algorithm found an  $\mathcal{RP}_{opt}$  matching, we only look at problems where (a) more than one stable matching exists (if only one stable matching exists, that solution is automatically an  $\mathcal{RP}_{opt}$  matching) and (b) the DA algorithm found a matching (the DA algorithms have different failure rates and we wanted to directly compare only when they succeed.) Table 5.4 also shows the general failure rate for the DA algorithms (i.e., when at least one stable matching exists, but the algorithm fails to find one).

Critically, note that even though the DA algorithms were designed to be as resident favoring as possible, they do not find an  $\mathcal{RP}_{opt}$  matching in a large percentage of problems with multiple stable matchings. When the percentage of couples is low, the DA algorithms’ performance is either perfect, or fairly good (with about a 5% failure rate). For problems with roughly the same percentage of couples as the NRMP (roughly 6%) the failure rate of KPR and RP99 to find a  $\mathcal{RP}_{opt}$  matching is fairly low, at 5% (though KPR’s failure rate is slightly higher.) However, with more couples in the match, the failure rate jumps to roughly 10%. Again, in all of these settings, SAT-E with the additional constraints was able to enumerate *all* stable matchings, including all  $\mathcal{RP}_{opt}$  matchings.

It appears as though RP99 finds an  $\mathcal{RP}_{opt}$  matching with

a slightly higher rate than KPR when it finishes. These two algorithms deal with couples fairly differently (as RP99 has couples propose individually, and couples propose in-batch in KPR), which could account for the difference in performance. However, KPR outperforms RP99 by a large margin with respect to the percentage of problem instances solved. KPR’s failure rate is less than 0.25% for all problems with 10% or less couples, and only 5.75% when the match has 20% couples. RP99’s failure rate grows quickly, reaching 36.25% by the time couples consist of 20% of the market. Therefore, it seems as though there is a slight benefit to using RP99 to find  $\mathcal{RP}_{opt}$  matchings, but its large failure rate may negate this benefit.

## 6 Discussion and Future Work

In this paper, we have provided new theoretical results on strategy-proofness for SMP-C, and new extensions to an existing SAT encoding for SMP-C that allows us to both enumerate every stable matching for a given problem instance, and guarantee that we’ve found a  $\mathcal{RP}_{opt}$  matching if we don’t have the resources to enumerate all instances. We then used these techniques to explore properties of SMP-C instances.

We used these tools to empirically investigate previous theoretical results. Interesting, and perhaps unexpectedly, the Immorlica and Mahdian results (given a constant list size, the probability of a unique stable matching goes to one as the market size increases) did not seem to hold. However, we empirically found evidence for a related phenomenon for SMP-C, leading us to the following conjecture: for problem instances drawn from the same distribution with a fixed ROL size of  $k$ , the probability of a resident optimal stable matching goes to one as the size of the market goes to infinity.

We also empirically evaluated RP99 and KPR, two DA-style algorithms for SMP-C that do not have any theoretical guarantees. While they frequently perform well on the problem instances we drew, we found many problem instances where they either did not find a stable matching at all even though one existed, or if they did find one, they did not find a  $\mathcal{RP}_{opt}$  matching. In practice, it would be a simple to use our algorithm 2 to improve the matching found by these algorithms.

By enumerating all possible stable matchings (and thus all resident Pareto optimal matchings for a given matching), we can now pick stable matchings based on certain properties. Enumerating the set of all stable matchings allows us to pick the matching with the best average rank for all singles and couples (a sort of resident social-welfare optimal matching), or the stable matching that places the worst-off individual as high in their preference rankings as possible. While we did not empirically evaluate this in this paper, being able to enumerate all stable matchings allows for a rich investigation into this problem.

Our overall conclusion is that encoding to SAT is a powerful tool for further empirical analysis of stable matching problems. The research question that arises is how best to use this tool and how this tool can be adapted to obtain insight into other important open questions about NP-complete stable matching problems. The current paper has shown that

this tool has already exposed some interesting theoretical questions. For example, what can be proved about the frequency in which resident-optimal matchings appear in randomly drawn SMP-C instances as the market size increases; is there further structure to the set of  $\mathcal{RP}_{opt}$  solutions; can the existence of a resident-optimal matching or conditions on the set of  $\mathcal{RP}_{opt}$  solutions be used to prove further strategy-proof results for SMP-C.

In addition to theoretical results, there is considerable potential for practical computation of stable matchings. For example, using both DA algorithms, and algorithm 2 to improve the resulting matching might have some very useful applications. It might also be the case that solving SAT-E with additional constraints might be applicable to some currently unsolved practical matching problems.

**Acknowledgements.** This work is supported by the Natural Sciences and Engineering Research Council of Canada. Per-rault and Drummond are additionally supported by an Ontario Graduate Scholarship.

## References

- [Abdulkadiroglu *et al.*, 2005] Atila Abdulkadiroglu, P.A. Pathak, Alvin E. Roth, and Tayfun Sönmez. The Boston public school match. *American Economic Review*, 95(2):368–371, 2005.
- [Ashlagi *et al.*, 2014] I. Ashlagi, M. Braverman, and A. Hasidim. Stability in large matching markets with complementarities. *Operations Research*, 62(4):713–732, 2014.
- [Biere, 2013] Armin Biere. Lingeling, plingeling and treengeling entering the sat competition 2013. In *Proceedings of the SAT Competition 2013*, pages 51–52, 2013. <http://www.satcompetition.org/2013/proceedings.shtml>.
- [Drummond *et al.*, 2015] Joanna Drummond, Andrew Perrault, and Fahiem Bacchus. SAT is an effective and complete method for solving stable matching problems with couples. In *Proc. of the Twenty-fourth International Joint Conference on Artificial Intelligence (IJCAI-15)*, 2015. To Appear.
- [Gale and Shapley, 1962] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69(1):9–15, Jan 1962.
- [Gelain *et al.*, 2010] Mirco Gelain, Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Local search for stable marriage problems with ties and incomplete lists. In *Proceedings of the 11th Pacific Rim International Conference on Trends in Artificial Intelligence (PRICAI-10)*, pages 64–75, Daegu, Korea, 2010.
- [Gent and Prosser, 2002] Ian P Gent and Patrick Prosser. SAT encodings of the stable marriage problem with ties and incomplete lists. In *Proceedings of Theory and Applications of Satisfiability Testing (SAT)*, 2002.
- [Gent *et al.*, 2001] Ian P Gent, Robert W Irving, David F Manlove, Patrick Prosser, and Barbara M Smith. A constraint programming approach to the stable marriage problem. In *Principles and Practice of Constraint Programming (CP)*, pages 225–239, 2001.
- [Immorlica and Mahdian, 2005] Nicole Immorlica and Mohammad Mahdian. Marriage, honesty, and stability. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 53–62, 2005.
- [Knuth, 1976] Donald E. Knuth. *Marriages stables*. Les Presses de l’Université de Montreal, 1976.
- [Kojima *et al.*, 2013] F. Kojima, P. Pathak, and A. E. Roth. Matching with couples: Stability and incentives in large markets. *Quarterly Journal of Economics*, 128(4):1585–1632, 2013.
- [Marx and Schlotter, 2011] Dániel Marx and Ildikó Schlotter. Stable assignment with couples: Parameterized complexity and local search. *Discrete Optimization*, 8(1):25–40, 2011.
- [Niederle *et al.*, 2008] Muriel Niederle, Alvin E. Roth, and Tayfun Sonmez. Matching and market design. In Steven N. Durlauf and Lawrence E. Blume, editors, *The New Palgrave Dictionary of Economics (2nd Ed.)*, volume 5, pages 436–445. Palgrave Macmillan, Cambridge, 2008.
- [NRMP, 2013] NRMP. National resident matching program, results and data: 2013 main residency match, 2013.
- [Ronn, 1990] Eytan Ronn. NP-complete stable matching problems. *Journal of Algorithms*, 11(2):285–304, 1990.
- [Roth and Peranson, 1999] A. E. Roth and E. Peranson. The redesign of the matching market for American physicians: Some engineering aspects of economic design. *The American Economic Review*, 89(1):748–780, September 1999.
- [Roth and Sotomayor, 1992] Alvin E. Roth and Marilda Sotomayor. Chapter 16. two-sided matching. In R. J. Aumann and S. Hart, editors, *Handbook of Game Theory Volume 1*, pages 485–541. Elsevier, 1992.
- [Roth, 1984] Alvin E. Roth. The evolution of the labor market for medical interns and residents: A case study in game theory. *Journal of Political Economy*, 92(6):991–1016, 1984.
- [Unsworth and Prosser, 2005] Chris Unsworth and Patrick Prosser. A specialised binary constraint for the stable marriage problem. In *Abstraction, Reformulation and Approximation (SARA)*, pages 218–233, 2005.

## A SAT Encoding for the Stable Matching Problem with Couples

In this appendix we provide an encoding of SMP-C into SAT. SMP-C was shown to be an NP-Complete problem by Ronn [Ronn, 1990]. Hence, it can be encoded and solved as a satisfiability problem (SAT). This is an attractive approach for solving SMP-C due to the recent tremendous advances in SAT solvers.

The encoding of SMP-C we give here forms the basis of Drummond et al. [2015]. Nevertheless, since we extend this SAT encoding in order to find Pareto optimal matchings and to find all stable matchings, we provide its details in this appendix (which will only be available on-line).

First we assume that the doctor (singles and couples) *rols* have been preprocessed so as to remove from them any program that does not find that doctor acceptable. For  $d \in S$  we remove  $p$  from  $rol_d$  if  $d \notin \text{rol}_p$ . For couple  $(d_1, d_2) \in C$  we remove  $(p_1, p_2)$  from  $rol_{(d_1, d_2)}$  if  $d_1 \notin \text{rol}_{p_1}$  or  $d_2 \notin \text{rol}_{p_2}$ . This ensures that in any matching found by solving the SAT encoding no doctor  $d$  can be matched into a program that does not rank  $d$ .

A SAT encoding consists of a set of Boolean variables and a collection of clauses over these variables. We first describe the set of variables and their intended meaning, and then we present the clauses which enforce the conditions required for a stable matching.

### A.1 Variables

We utilize three sets of Boolean variables.

#### 1. Doctor Matching Variables:

$$\{m_d[p] \mid d \in D \wedge p \in \text{rol}_d\}$$

$m_d[p]$  is true iff  $d$  is matched into program  $p$ . Note that  $m_d[\text{nil}]$  is true if  $d$  is unmatched.

#### 2. Couple Matching Variables:

$$\{m_c[i] \mid c \in C \wedge (0 \leq i < |\text{rol}_c|)\}$$

$m_c[i]$  is true iff couple  $c$  is matched into a program pair  $(p, p')$  that it ranks between 0 and  $i$ , i.e.,  $0 \leq \text{rank}_c((p, p')) \leq i$ . (Lower ranks are more preferred).

#### 3. Program Matching Variables:

$$\{m_p[i, s] \mid p \in P \wedge (0 \leq i \leq |\text{rol}_p| - 2) \wedge (0 \leq s \leq \min(i + 1, \text{cap}_p + 1))\}$$

$m_p[i, s]$  is true iff  $s$  of the doctors in  $\text{rol}_p[0]$  to  $\text{rol}_p[i]$  have been matched into  $p$ . Note that  $i$  ranges up to  $|\text{rol}_p| - 2$  which is the index the last non-*nil* doctors on  $\text{rol}_p$  ( $\text{rol}_p$  is terminated by *nil*).

The program matching variables are the main innovation of our encoding. These variables are especially designed to allow us express more efficiently the constraints a stable matching must satisfy.

### A.2 Clauses

Now we give the **clauses** of the encoding. Rather than give the more lengthy clauses directly, we often give higher level constraints whose CNF encoding is straightforward.

**Unique Match** A doctor must be matched into exactly one program (possibly the *nil* program). For all  $d \in D$

$$\mathbf{1a} \text{ at-most-one}(\{m_d[p] \mid p \in \text{ranked}(d)\})$$

$$\mathbf{1b} \bigvee_{p \in \text{ranked}(d)} m_d[p]$$

The **at-most-one** constraint simply states that at most one of a set of Boolean variables is allowed to be true. It can be converted to CNF in a number of different ways. **1b** ensures that some match (possibly to the *nil* program) is made.

**Couple Match** The  $m_c[*]$  variables must have their intended meaning. For all couples  $c \in C$ , for all  $k$  such that  $1 \leq k \leq |\text{rol}_c|$ , letting  $c = (d_1, d_2)$  and  $(p_1[i], p_2[i]) = \text{rol}_c[i]$ ,

$$\mathbf{2a} \ m_c[0] \equiv m_{d_1}[p_1[0]] \wedge m_{d_2}[p_2[0]]$$

$$\mathbf{2b} \ m_c[k] \equiv (m_{d_1}[p_1[k]] \wedge m_{d_2}[p_2[k]]) \vee m_c[k - 1]$$

$$\mathbf{2c} \ m_c[|\text{rol}_c|]$$

The final condition ensures that  $c$  is matched to some program pair on its *rol* (possibly *nil*), and the **at-most-one** constraint for  $d_1$  and  $d_2$  ensures that  $c$  is uniquely matched.

**Program Match** The  $m_p[*]$  variables must have their intended meaning. For all programs  $p \in P$ , for all  $i$  such that  $1 \leq i \leq |\text{rol}_p| - 2$ , and for all  $s$  such that  $0 \leq s \leq \min(i + 1, \text{cap}_p + 1)$ , letting  $d_i = \text{rol}_p[i]$  (the  $i$ -th doctor on  $p$ 's *rol*),

$$\mathbf{3a} \ (m_p[0, 0] \equiv \neg m_{d_0}[p]) \wedge (m_p[0, 1] \equiv m_{d_0}[p])$$

$$\mathbf{3b} \ m_p[i, s] \equiv (m_p[i - 1, s] \wedge \neg m_{d_i}[p]) \vee (m_p[i - 1, s - 1] \wedge m_{d_i}[p])$$

$$\mathbf{3c} \ \neg m_p[i, \text{cap}_p + 1]$$

The last condition, captured by a set of unit clauses, ensures that  $p$ 's quota is not exceeded at any stage. Falsifying these variables along with the other clauses ensures that no more matches can be made into  $p$  once  $p$  hits its quota. Note that  $i$  only indexes up to the last non-*nil* doctor on  $\text{rol}_p$  since *nil* does not use up any program capacity.

**Stability for Singles** For each single doctor,  $d \in S$  and for each  $p \in \text{rol}_d$

$$\mathbf{4} \ (\bigvee_{p' \succeq_d p} m_d[p']) \vee m_p[\text{rank}_p(d) - 1, \text{cap}_p]$$

This clause says that if  $d$  has not been matched into a program preferred to or equal to  $p$ , then it must be the case that  $p$  will not accept  $d$ . Note that  $m_p[\text{rank}_p(d) - 1, \text{cap}_p]$  means that  $p$  has been filled to capacity with doctors coming before  $d$  on its *rol*.

**Stability for Couples (A)** For each couple  $c = (d_1, d_2) \in C$  and for each  $(p_1, p_2) \in \text{rol}_c$  with  $\mathbf{p_1} \neq \mathbf{p_2}$

$$\mathbf{5a1} \ m_{d_1}[p_1] \wedge \neg m_c[\text{rank}_c((p_1, p_2))] \implies m_{p_2}[\text{rank}_{p_2}(d_2) - 1, \text{cap}_{p_2}]$$

$$\mathbf{5a2} \ m_{d_2}[p_2] \wedge \neg m_c[\text{rank}_c((p_1, p_2))] \implies m_{p_1}[\text{rank}_{p_1}(d_1) - 1, \text{cap}_{p_1}]$$

$$\mathbf{5b} \ \neg m_{d_1}[p_1] \wedge \neg m_{d_2}[p_2] \wedge \neg m_c[\text{rank}_c((p_1, p_2))] \implies m_{p_1}[\text{rank}_{p_1}(d_1) - 1, \text{cap}_{p_1}] \vee m_{p_2}[\text{rank}_{p_2}(d_2) - 1, \text{cap}_{p_2}]$$

Clause **5a1** says that when  $d_1$  is already matched to  $p_1$  but  $c$  has not been matched into  $(p_1, p_2)$  or into a more preferred program pair, then it must be the case that  $p_2$  will not accept  $d_2$ . **5a2** is analogous.

Clause **5b** says that if neither  $d_1$  nor  $d_2$  is matched into  $p_1$  or  $p_2$  and  $c$  has not been matched into  $(p_1, p_2)$  or into a more preferred program pair, then either  $p_1$  will not accept  $d_1$  or  $p_2$  will not accept  $d_2$ .

**Stability for Couples (B)** For each couple  $c = (d_1, d_2) \in C$  and for each  $(p, p) \in \text{rol}_c$  we have one of constraint **6a1** or **6b1**. **6a1** is needed when  $d_1 \succ_p d_2$ , while **6b1** is needed when  $d_2 \succ_p d_1$ .

$$\mathbf{6a1} \quad m_{d_1}[p] \wedge \neg m_c[\text{rank}_c((p, p))] \implies m_p[\text{rank}_p(d_2) - 1, \text{cap}_p]$$

$$\mathbf{6b1} \quad m_{d_1}[p] \wedge \neg m_c[\text{rank}_c((p, p))] \implies m_p[\text{rank}_p(d_1) - 1, \text{cap}_p - 1]$$

$$\mathbf{6c} \quad \neg m_{d_1}[p] \wedge \neg m_{d_2}[p] \wedge \neg m_c[\text{rank}_c((p, p))] \implies m_p[\text{rank}_p(d_1) - 1, \text{cap}_p] \vee m_p[\text{rank}_p(d_1) - 1, \text{cap}_p - 1] \vee m_p[\text{rank}_p(d_2) - 1, \text{cap}_p] \vee m_p[\text{rank}_p(d_2) - 1, \text{cap}_p - 1]$$

Clauses **6a1** or **6b1** say that if  $d_1$  is already in  $p$  and  $c$  is not matched to  $(p, p)$  or into a more preferred program pair, then  $p$  will not accept  $d_2$ . **6b1** differs because when  $d_2 \succ_p d_1$  and  $d_1$  is already in  $p$ ,  $p$  will definitely accept  $d_2$ . In this case, however, the couple is not accepted into  $(p, p)$  if accepting  $d_2$  causes  $d_1$  to be bumped. That is, when  $m_p[\text{rank}_p(d_1) - 1, \text{cap}_p - 1]$  is true (adding  $d_2$  will cause  $m_p[\text{rank}_p(d_1) - 1, \text{cap}_p]$  to become true).

There are also analogous clauses **6a2** and **6b2** (one of which is used) to deal with the case when  $d_2$  is already in  $p$  and we need to ensure that  $p$  won't accept  $d_1$ . Clause **6c** handles the case when neither member of the couple is currently matched into  $p$ .

Let  $\mathcal{P} = \langle D, C, P, \succeq \rangle$  be a matching problem. We say that a matching  $\mu$  for  $\mathcal{P}$  and a truth assignment  $\pi$  for SAT-E of  $\mathcal{P}$  are **corresponding** when  $\pi \models m_d[p]$  iff  $\mu(d) = p$ .

We provide a full proof of the following theorem in Drummond et al. [2015].

**Theorem 2.** *If  $\mu$  and  $\pi$  are corresponding, then  $\mu$  is stable if and only if  $\pi$  is satisfying.*

This theorem shows that the satisfying assignments of SAT-E are in a 1-1 relationship with the stable models.